AMIGA WORKBENCH

$2 RRP

Registered by Australia Post – Publication No. VBG7930

Number 22 | Circulation: 1000 | March 1988

Next AUG Meeting

Saturday, March 12th, 1988 at 2pm

(Doors open at 1pm, meeting starts at 2pm sharp)

AUG meetings are held in the Rotunda at Monash University
Wellington Road, Clayton Melways map 70 reference F10 and map 84A

Amiga Users Group inc, PO Box 48, Boronia, 3155, Victoria, Australia

Australia's Largest Independent Association of Amiga Owners
The *Amiga Users Group Inc* has no affiliation with Commodore

Who Are We?

The Amiga Users Group is a non-profit association of people interested in the Amiga computer and related topics. With almost 900 members, we are the largest independent association of Amiga users in Australia.

Club Meetings

Club meetings are held at 2pm on the meeting day in the Rotunda at Monash University, Wellington Road, Clayton. Details on how to get there are on the back cover of this newsletter. The dates of the next few meetings are:

Saturday, March 12th at 2pm
Sunday, April 17th at 2pm
Sunday, May 15th at 2pm

Production Credits

This month's newsletter was edited by Peter Jetson. Equipment and software used was: TurboDOS S-100 computer, Brother HR-40 printer, Gemini 10x printer, Wordstar, Fancy Font and Grabbit.

Copyright and Reprint Privileges

Contributions

Articles, papers, letters, drawings and cartoons are actively sought for publication in Amiga Workbench. Please submit your contributions on disk, since that means they don't have to be re-typed! All disks will be returned! Please save your article in **text-only** format (If it can be loaded by ED, it is text-only). Absolute deadline for articles is 16 days before the meeting date. Contributions can be sent to: The Editor, AUG, PO Box 48, Boronia, 3155.

Membership and Subscriptions

Membership of the Amiga Users Group is available for an annual fee of $20. To become a member of AUG, fill in the membership form in this issue (or a photocopy of it), and send it with a cheque for $20 to:

Amiga Users Group, PO Box 48, Boronia, 3155

Public Domain Software

Disks from our public domain library are available on quality 3.5" disks for $8 each including postage on AUG supplied disks, or $2 each on your own disks. The group currently holds over 160 volumes, mostly sourced from the USA, with more on the way each month. Details of latest releases are printed in this newsletter, and a catalog disk is available.

Member's Discounts

The **Amiga Users Group** negotiates discounts for its members on hardware, software and books.

Currently, **Technical Books** in Swanston Street in the city offers **AUG** members a 10% discount on computer related books, as does **McGills** in Elizabeth Street. Just show your membership card. Although we have no formal arrangements with other companies yet, most seem willing to offer a discount to AUG members. It always pays to ask!

Back Issues of Newsletter

All back issues of Amiga Workbench are now available, for $2 each including postage. Back Issues are also available at meetings.

AmigaLink - Our Bulletin Board System

The Amiga Users Group operates a bulletin board system devoted to the Amiga, using the Opus message and conferencing system. AmigaLink is available 24 hours a day on (03) 792 3918, and can be accessed at V21 (300bps), V22 (1200bps) or V23 (1200/75bps), using 8 data bits, 1 stop bit and no parity.

AmigaLink is part of the world-wide Fido/Opus network of bulletin boards, and we participate in the national and international Amiga conferences. AmigaLink has selected Public Domain software available for downloading, and encourages the uploading of useful public domain programs from its users. AmigaLink is FidoNet node number 631/324.

Newsletter Advertising

The Amiga Users Group accepts commercial advertising in Amiga Workbench subject to the availability of space at these rates:

Quarter page	$20
Half page	$40
Full page	$70
Double page spread	$120

These rates are for full-size camera-ready copy **only**. We have no photographic or typesetting facilities. Absolute deadline for copy is 16 days before the meeting date. Send the copy and your cheque to: The Editor, AUG, PO Box 48, Boronia, 3155, Victoria.

Amiga Users Group Committee

Game Review: Fire Power

by Stan Thomas

It didn't take long to be briefed on my mission: capture the enemy's flag. I chose one of the three tanks on offer, the Scorpion, a massive and powerful piece of weapons technology, and a good one for what was my first mission. I drove it out of the garage onto the roads winding through the fortress and, without stopping to think of the terrors and dangers ahead, rolled out through the gate and into enemy territory.

Suddenly my tank was rocked with the force of an explosion. The roads were mined! I knew what I had to do if I wanted to survive. I turned south and began firing at the fortress wall blocking my way.

In moments, part of the wall disintegrated in a blast of fire and I headed through the gap onto the safer grasslands. I had no idea at this stage where to start looking for the enemy flag, and so I continued south until I felt myself bombarded once more. A gun turret in the forest was firing at me. I could have ignored it and carried on, but, feeling I needed some target practice, I began firing back.

It didn't take long. Soon the gun was a smouldering ruin. I could see a couple of survivors running away. I knew I could go after them and get extra points for chasing them and turning them into a red stain on the ground, but decided not to. After all, I had more important things to do.

Then, in the distance I heard the familiar thwack thwack of the helicopter. I had stayed around too long and now they had a bearing on me.

I turned and headed south again, searching for that flag, as the helicopter fired a missile at me, damaging my tank badly. I stopped, turned, fired. The helicopter turned into a ball of fire and was gone.

Then, to my left was a whole battery of gun turrets, all seeking my destruction. I let rip, firing in all directions, until I had destroyed enough guns to make a safe entrance into their fortress.

Once inside, I saw a prison hut and fired on it. Once it exploded, I stood still, dangerously still, while our soldiers, held prisoner for who knows how long, ran towards my tank. Then I turned back the way I came, intent on finding a first aid hut where I could drop my cargo of rescued prisoners.

Suddenly, without warning, everything froze. The sound of explosions began repeating themselves, as if stuck in an endless loop. Nothing moved, not my tank, not the gun turrets, not the helicopter that was bearing down on me.

Everything went black and silent. Then a message appeared, starting with the words 'Guru Meditation'. Yes, folks, each and every time I play the game 'Fire Power' from Microillusions I get message number 81000005 from the Guru, telling me that the memory list has become corrupted. How is it that a piece of software can be allowed on the market with what are obviously major bugs? This is a very great pity, since it is basically quite a good game, with simple but effective graphics that consist of a very large scrolling area made up, as far as I can tell, of a set of bobs used in the way that custom character sets are used on the Commodore 64 to obtain a very large picture. Limited but effective digitised sounds (this is becoming routine - what game on the Amiga doesn't have digitised sound now?) consists of tank, helicopter and explosion noises.

A feature of this game is its choice of playing mode. The first, and the one that attracted the Guru, is the player against the computer. The second is player against player. The third is rather novel, and allows you to play against someone else using a modem. Since I haven't used either of these last two modes, I can't say whether they will crash or not.

Another element in this game I dislike, especially these days when people are concerned about excessive violence in the media, is the way points can be earned by running fleeing soldiers over with your tank. A small red splotch appears on the screen when you do this. This to me is an unnecessary and gratuitous addition to an otherwise interesting game.

In summary, it is a fairly good game that I cannot recommend until the bugs that cause it to crash so dependably have been fixed.

===

RAMJET 1

EXTERNAL EXPANSION MEMORY FOR COMMODORE AMIGA™

FEATURES

- 1 megabyte of FAST (no wait states) memory
- Low cost - only $495
- Compatible with AMIGA 1000 and AMIGA 500
- Pass through bus
- Dimensions: Height 120mm, Length 205mm, Width 30mm
- 6 month warranty

The RAMJET 1 is made in Australia by:
Control Technology Pty. Ltd.
Telephone: (03) 729-4272

™COMMODORE AMIGA is a trademark of Commodore Business Machines

Dear Editor

This is in response to your request for articles about user's experiences and purchases. I have yet to invest the $600+ for the privilege of programming in C and for those expensive reference manuals. However the programs in the Basic Demo drawer proved to be excellent tutorials in the use of libraries and how to access the screen data directly through PEEKS and POKES. I have been using Amiga Basic as a friendly interface with the Amiga's operating system and have concentrated on learning and using the 68000 instruction set. Assembly listings such as those in Enno Davids article are much appreciated as they elucidate the practical side of programming the Amiga.

I purchased the Amiga 1000, DigiView and video camera with the purpose of writing image processing and recognition algorithms- the reason for needing machine code. The DigiView software is easy to use and does what it is surposed to do- put the prettiest picture possible into your Amiga using any of the display modes that your computer's memory can handle.

Unfortunately this wasn't all I wanted to do and the manual did not reveal the circuit used nor provide any software that would enable it to be used to capture images under control of your own software. So for the time being I have to manually store the images onto a disk and let the software pretend that it is getting the data from the camera.

My other big purchase was the KickStart Eliminator. The extra 256k allows me to digitise all but the highest resolution and, despite the cost, it has been worth having.

-- from J.G. Casey

==

Introduction to the Programming Language `C'
Part 2 - The Environment
by Eric Salter

The key to successful `C' development on the Amiga is to set up your compiler environment correctly. Errors caused by the compiler or linker not being able to find either a header file or library are no help when you are trying to get a program going and only add 'noise' to your own program's error messages. In the end, you don't know if the errors were caused by your program as such, or as a result of a varible or member of a structure not being defined because a header file was not found or read in the wrong order. Getting the environment right is the single most productivity minded thing you can do and a little time carefully designing this environment can save you hours of fruitless debugging time.

Hardware requirements

There are a number of system resources we have to make efficient use of when we compile. The most important resource is memory and compiling requires vast inhuman quantities the stuff. The need for large amounts of memory is due partly to the size of the compiler itself which lists at 91,988 bytes for the first pass and 88,400 bytes for the second in Lattice 3.10. The compiler then has to have room to read in your source (although it operates on the stream rather than pulling the entire file in at once) and generate its own tables and produce intermediate files. Compiling also takes its toll on the stack and a realistic stack size to compile any reasonably sized `C' program is 15,000 bytes. Now, if all that was in memory was the operating system and the compiler, it would be possible to compile with a standard 512K Amiga, BUT this is not a realistic option as we will see in a moment. I personally believe that 'C' development only becomes most practical and efficient on a machine with over 2 Megabytes of memory.

The second most important consideration is disk-space. There are a large number of files involved in the `C' environment. These include: The compiler itself, the linker, the "include" files to support our programming environment, the libraries that are scanned by the linker, and the AmigaDOS environment, all the AmigaDOS CLI commands. This alone comes to well over 1.3 Megabytes of files and obviously cannot fit onto one 880 Kbyte floppy and we haven't included all the other programs you can't do without! You cannot successfully compile without at least 2 floppy disk drives as a MINIMUM! For those who can afford it, a hard disk is the most efficient means of storage and I would recommend one to anyone doing serious `C' development. For those without a hard disk, the use of a recoverable ram-disk will give you extra storage and speed and enable you to recover after a crash.

The biggest disadvantage to using floppies to compile with is their slowness. Consider what happens when you compile: The compiler is loaded off disk (the lc at 25 K) which interprets the command-line you passed to it. It invokes the first pass of the compiler (all 92 K) which reads your source code in, pausing to read in the "include" files you specified, generates an intermediate file while it's reading your source file. The second pass of the compiler is now loaded (89 K) and proceeds to produce an object file from the intermediate file left by the first pass. The linker is now loaded (35 K) and reads the object file and searches the libraries (89 K and 79 K minimum) and produces the load-file and map-file. This amounts to a lot of disk activity colourfully described as shoe-shining, after the way in which the heads seek from one side of the disk to the other as the various files are read in and created simultaneously (i.e. read a bit/ write a bit). Not only does it wear out your disk heads, but it wears out your patience with loooong compiles and links. It wouldn't be so bad if your code was error free, but when you get an error after 25 minutes of disk-thrashing, your Amiga's screen is in grave danger from swinging axes!

You can reduce dramatically the amount of time necessary for compiling by doing most of it in a Ram-disk and this is where the 2 Megabytes comes in handy. If you only have a floppy drive, the only disk access you should be doing is reading the source file, the "include" files and writing out the object and load files. A hard disk makes life so much easier.

Apart from hardware configuration, the programming environment is the key to doing it easily. With each

compile and link, we have to be able to find several things:

Lc, Lc1, Lc2	The Lattice 3.10 compiler
Include files	stdio.h, exec/types.h etc...
The QUAD file	The first-pass intermediate file
The object file	Output from the second pass
Blink	The linker
c.o	The startup code (like lstartup.o in 3.03)
lc.lib, amiga.lib, etc	The scanned libraries

If you are working on a floppy-only environment, it is a good idea to boot up from a disk which will set up the right assignments automatically. Here is an example of the startup-sequence from such a disk. It uses the public domain programs "CONMAN" and the "ASDG Recoverable Ram-disk" as well as a proprietary program to read my system's real-time clock. This system has 1.5 Megabytes of Ram and 3 disk-drives:

```
run conman >nil: -q
decigel
newcli con:0/0/640/256/C_Development from s/old
endcli >nil:
```

The startup sequence above runs CONMAN quietly, sets up a 68010 illegal instruction processor (I use a 68010), then opens a new CLI full screen and causes the new CLI to read the file "old" in the "s" directory as its input - in effect opening a new CLI and typing "execute s/old" in its window. The last line terminates the initial CLI. It is necessary to open a new CLI to reap the benefits of CONMAN which nobody should be without. Now the file s/old looks like:

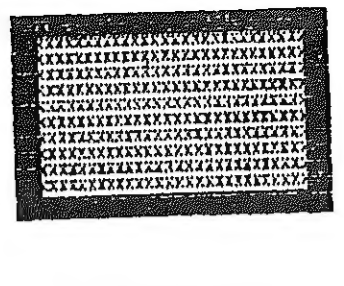
```
stack 15000                  ;give us room to compile
addbuffers df0: 15           ;more buffers
addbuffers df1: 15
addbuffers df2: 15

echo "C Development Disk Lattice 3.10"
echo

RTClock -q                   ;read the clock

echo "System Time: "
date
echo
echo "Making Ram-disk for compiles"
echo
mount VDO:                   ;mount the ASDG Ram-disk
if not exists VDO:libs       ;if not there make it
 makedir VDO:lib
 copy lib to VDO:lib all quiet
endif
if not exists VDO:libs/amiga.lib
 copy lib to VDO:lib all quiet
endif
makedir RAM:c
copy c: to RAM:c all quiet

assign C: RAM:c
assign LC: C:
assign INCLUDE: INC:
assign LIB: VDO:lib
assign QUAD: RAM:
cd df2:
```

To explain the above: The ASDG Ram-disk is called VDO: and has to be mounted on the system device list. The mount command will search the "mountlist" in the DEVS: directory to find out what is needed and where to load the handler etc. We then test to see if the VDO:lib is in existence. If it is (i.e. as a result of a previous boot - it is recoverable you know) then we don't have to waste time re-loading the libraries. We test to see if the libraries are actually there (in case the original boot process was interrupted and there are no libraries in the directory) and we copy all the C: directory files (which includes the compiler) into the Ram:c. It would be nice to load them into the Recoverable disk but they won't all fit on a 1.5 Megabyte machine (ASDG is set up for 512 K max) so we have to use the RAM: device. The assignments are the most critical parts of this sequence. They tell the compiler where to find its various bits. Under Lattice, there are a number of logical names used to find these bits:

LC	This is used by "Lc" to find the fist and second parts of the compiler. Lc might do something like: LC:lc1 myprogram.c. It doesn't matter where lc1 is as long as we have set up the assignment of LC: correctly, the program will find it. The same is true for:
INCLUDE	Where to find the "include" files. In this case, they are found on the volume called INC: (which may be on any drive but the system will find it).

COMACC

SCREEN ENHANCER



REDUCES HI-RES JITTER
REDUCES EYESTRAIN AND HEADACHES
IMPROVES CLARITY AND CONTRAST
ENHANCES COLOURS

SUITS COMMODORE MODELS 1081 & 1084

FITS EASILY - VELCRO PADS PROVIDED

$20

SEND COUPON TO:-
COMACC
1813 ALBANY HIGHWAY
MADDINGTON, W.A. 6109
Phone 09 459-5523

- -

Please send SCREEN ENHANCER/s.
I enclose cheque/money order $.....
Name.................................
Address..............................
....
State...............P.Code......

LIB	The libraries - found in the VDO:lib directory
QUAD	Where to find the intermediate quad file.
C	Where to find the commands like lc.

The compiler and linker use the logical names for the place to find the libraries and "include" files etc. In this manner, we can operate on a wide variety of physical hardware configurations and merely by specifying the correct assignments, the compiler can find all it needs. Finally, we have set the current directory to DF2:. It is here we will put our source-code and write out our object and load files. With 1.5 Megs of memory, we can't afford to keep all our "include" files in Ram as well, so we keep them on a seperate disk which we name "INC". To set up for a hard-disk-based development environment, it is a simple matter of changing the assignments to where you keep the various bits on the hard-disk.

Next time, we will look at each of the tools in the C compiler and look at their function. After we get to know our tools, we will get into the C language in earnest.

==

The Advanced C Special Interest Group

by Eric Salter

This new SIG met for the first time at the meeting in February at Monash. We had about 8 people turn up mainly because it was an impromptu meeting, decided upon at the time. Anyway, it should now be a permanent fixture to our meetings.

At this meeting, we looked at the organization of the operating system which is the first step to writing intelligent code for the amiga. We looked at libraries, concentrating on the exec and its functions and how to use it to write code that behaves properly in a multi-tasking environment. A short introduction to using system data structures and handling lists was one of the last items discussed before time, and my voice, ran out.

Next time, we will look more formally at traversing system lists and how to make sure they are consistent by preventing multi-tasking and/or interrupts.

We are looking out for a place to hold some C courses. These courses would run over about 8 weeks one evening a week and cover C on the Amiga. Please, if anyone can help, we need a venue with room for about 10 people and computers (BYO computer), contact me on (AH) 861-9117 or (BH) 418 2211.

==

AmigaLink

300, 1200 & 1200/75 bps

(03) 792 3918

New Into the AUG Public Domain Software Library

Amicus #21

This disk has electronic catalogs for the Amicus disks 1 thru 20 and Fish disks 1 thru 80. They can be viewed with the DiskCat program, also included on this disk.

Target	Makes each mouse click sound like a gunshot
Sand	Simple "game" of sand that follows the mouse pointer around the screen
PropGadget	Proportional gadget example
EHB	Checks to see if you have extra-half-bright graphics on your Amiga
Piano	Simple piano sound program
CelScripts	Makes Cel animation scripts for Aegis Animator, in AmigaBASIC

Amicus #22

Cycles	Light cycle game
Show_PrintII	Views and prints IFF pictures, including larger than screen
PrtDrvGen2.3	Latest version of a Printer Driver generator
Animations	VideoScape animations of planes and boing ball
Garden	Makes fractal landscapes
BasicSorts	Examples of binary search and insertion sort in AmigaBASIC

Amicus #23

This disk is completely dedicated to music on the Amiga, and contains two music players, songs and instruments to bring the thrill of playing "Big Sound" to your Amiga.

Instruments	A collection of 25 instruments for playing and creating music, ranging from Cannon to Marimba
ListINSTR	Program to list the instruments DMCS will not load as well as list the origins for any instrument
Music	A collection of 14 classical pieces
1812Overture	The 15 minute classical feature complete with cannon!

Amicus #24

The VirusCheck directory holds several programs relating to the software virus that came to the US from pirates in Europe, as detailed in Amazing Computing magazine, V2.12. Bill Koester's (CATS) full explanation of the virus code is included. One program checks for the software virus on a WorkBench disk, the second program checks for the virus in memory, which could infect other disks.

Sectorama	A disk sector editor for any AmigaDOS file-structured device. Recover files from a hard disk
Iconize	Reduces the size of IFF images. Companion program, Recolor, remaps the palette colors of one picture to use the palette colors of another. Using these programs and a tool to convert IFF brushes to Workbench icons, you can make icons that are miniatures of screen pictures
CodeDemo	Modula-2 program converts assembler code object files to inline CODE statements. Comes with a screen scrolling example
AmiBug	Workbench hack makes a fly walk across the screen at random intervals
BNtools	Three examples of assembly language code: Setlace, a program to switch interlace mode on or off Why, a replacement for the AmigaDOS why command Loadit, loads a file into memory until reboot, probably only useful to real Amiga hackers.
Monolace	A CLI program to reset preferences to several colors of monochrome and interlace screens. C source included, works with DisplayPref, a CLI program which displays the current preferences settings
BoingMachine	A ray-traced animation of a perpetual-motion Boing-making machine, includes the latest version of the Movie program, which has the ability to play sounds along with the animation
Daisy	Example of using the translator and narrator devices to make the Amiga talk. Written in C
QuickFlix	Script-driven animation and slideshow program flips through IFF pictures
BMon	System monitor in AmigaBASIC, perform simple manipulations of memory
Moose	Random background program, a small window opens with a moose resembling Bullwinkle saying user-definable witty sayings
DGCS	Deluxe Grocery Construction Set, simple Intuition-based program for assembling and printing a grocery list

Amicus #25

Nemesis	Graphics demo pans through space towards a mythical dark twin of the sun. With wonderful music and space graphics
KickPlay	A text that describes several patches to the KickStart disk. This is for Amiga 1000 hackers who feel comfortable patching a disk in hexadecimal. KickPlay offers the chance to automatically do an ADDMEM for older expansion memory, as well as the ability to change the picture of the "Insert WorkBench" hand. A program is also included for restoring the correct checksum on a KickStart disk
KeyBird	BASIC program for editing keymaps
8ColorWB	Modifies WorkBench so three bitplanes are used, so icons can have 8 colors instead of 4. Sample 8 color icons are included, along with ZapIcon and Brush2Icon to convert 8 color brushes to Icons. Use Delux Paint to create your own 8 color icons
BrushIcon	Converts brushes to icons, with bizarre documentation
Egraph	Graphing program, reads (x,y) values from a file and displays them on the screen, similar to the unix program of the same name
Keep1.1	Message managing program for telecommunications. If you save messages from an online session to a file, Keep1.1 will let you move through the file one message at a time with the mouse. Keep1.1 understands the national (US) network message formats, and several types of bulletin board formats
Kill.fastdir	Speed up directory access. It creates a small file in each directory of a disk which contains information about the files in that directory. Also removes all those annoying "fastdir" files from a disk!
LaceWB	Changes between interlace and non-interlace WorkBench screen. Previously, you were forced to reboot after changing preferences to an interlaced screen. This program also flips between the normal and extended screen heights
PW_Utility	A shareware utility for ProWrite users, changes margin settings and font types
Guru	A CLI program that decodes the left side of a Guru Meditation number into English
DiskWipe	Removes files from directories or disk drives, much faster than delete

A quarter-page advert in

Amiga Workbench

will cost you only

$20

(from camera-ready copy)

Other sizes available are:

Half Page:	**$40**
Full Page:	**$70**
Double page:	**$120**

We can also insert your flyers (Contact us for rates)

See inside front page of this newsletter for more details

Snow	AmigaBASIC program makes snowflake-like patterns on the screen
Mlist	Mailing list database
SoftBallStats	Maintain softball statistics/team records
Dodge	Short Modula-2 program moves the WorkBench screen around after a period of time, prevents monitor burn-in

Amicus #26

This disk contains Todor Fay's SoundScape module code from his Amazing Computing articles. The source to Echo, Chord, TX and VU is included. The Lattice and Manx C source code is here, along with the executable modules.

ImageMaker	Interesting tool edits image structures for C, loads and saves C code directly
Claz2	Update of program to convert IFF images to postscriptfiles for printing on laser printers
SDBackup	Hard disk backup program with Lempel-Ziv compression to reduce the number of disks needed
TCB	Prints information about tasks and processes in the system; assembler source included
FunBut	Lets a function key act like a rapid series of left mouse button events, great for some games
DC	A handy program for people who use an Amiga 1020 5 1/4 inch drive as an Amiga floppy. A WorkBench program that sends a disk change signal to the operating system. Instead of typing "DiskChange df2:" over and over again, just click on the icon. C source included
SystemConfig	File makes screen 80 columns wide in the Scribble! word processor
Dic2Ram	2 programs to move the Scribble! spelling dictionary to and from the RAM disk
Lexical	Analyses a text file and gives the Gunning-Fog, Flesch and KinCaid indices which measure readability
HexDump	Modula-2 program to display memory locations in hexadecimal
Tartan	AmigaBASIC program to design tartan plaid patterns
DirMaster	Disk cataloging program
BMP	Plays 8SVX sampled sounds in the background while something else is happening in the Amiga; as your Amiga is booting for instance
ShowPt	CLI program changes your pointer, comes with a collection of pointers for you to use

==

Shell - A CLI Enhancement
by Lester McClure

Anyone who regularly uses the Amiga Command Line Interpreter (CLI) soon discovers its limitations and restrictions, especially compared with other computer systems. This is understandable because the CLI was never intended to be the prime user interface on the Amiga; Workbench is provided for that purpose. There are however, many reasons for using the CLI and some things simply cannot be achieved using Workbench.

The major shortcomings with the AmigaDOS CLI compared with systems such as UNIX and even MSDOS are:-

o None of the commands are built in to the interpreter. They all exist as disk files and have to be loaded before execution. This can be overcome to a certain extent by loading frequently used commands into RAM:

o There isn't any form of command history or command line editing. You have to re-type every command line in full although you may wish only to execute your last command again, perhaps with a simple change.

o The concept of redirection is supported but not 'pipes' where the output from one command becomes the input to another.

o Automatic execution of command or batch files is not possible. Explicit use of the Execute command is required. MSDOS uses special filename extensions (.BAT) and with UNIX all that is required is to set the file attributes appropriately.

o Filename expansion is done in a very non-standard way. The wildcard characters '*' and '?' have been with us for many years and there is no excuse for the Amiga not to support them.

o Function keys are not used with the CLI and the HELP key does not provoke any useful response.

There have been a number of attempts to overcome these problems and two from the Public Domain are worth considering.

The first is ConMan which is a replacement console handler which provides line editing and command line history. Once installed, any windows opened by AmigaDOS will automatically open using the ConMan handler. Previously entered command lines (to a limit of 10) can be recalled using the up-arrow, and the left and right arrows allow you to move along a command line. A command line can be edited by using Delete and Backspace and new characters can be inserted at the cursor position.

This overcomes the major objections I have to AmigaDOS CLI, however, another program does all this and more. It is called 'Csh' or 'Shell' and is an implementation of the UNIX C-Shell. The shell on a UNIX system surrounds the inner DOS and provides a user interface that is both a programming language and a command interpreter.

Matt Dillon originally implemented Csh for the Amiga (Fish Disk #14) and it has been enhanced more-recently by Steve Drew. I currently use V2.06 from Fish disk #85 but I believe V2.07 has been released on disk #107. Csh works under AmigaDos V1.2 only. It is approximately 35kbytes in size and would normally be placed in the C: directory of your system disk. It is completely in the Public Domain, is supplied with Manx C source and a 12 page documentation file describes the features and gives

simple examples of how to set it up and use it.

Shell Overview

The major features of Csh are described in this article. More details can be found in the supplied documentation files but the real advantages become obvious once you start using it.

o Built in commands - the most frequently used DOS commands are part of the shell and no longer need to be loaded from disk. This includes 'dir', 'cd', 'copy', 'run', 'cat' (Type), 'mv' (Rename) and 'rm' (Delete). Some of these are different from the original CLI commands but I consider the changes in general to be improvements. If the original command with the same name is required, simply type the name with the first letter in upper-case, e.g.

```
'Dir df0: opt I'.
```

o Command History and Editing is fully supported. The limit is specified by the user (defaults to 20), the complete stored history can be viewed, any past command can be directly executed or the up/down arrows can step back/forward through the list. A command line can be edited by using the left/right arrows and Delete/Backspace keys or any number of special control codes. This can be turned off if you are using a modified console handler such as ConMan.

o Pipes have been implemented via temporary RAM: files. These files have completely unique names and are automatically deleted on completion of the pipe segment. Input and output redirection are incorporated along with output append redirection i.e. '>>'. This specifies that the output from a command is to be appended to the end of an existing file. This can be useful for creating a system log file e.g. 'date >>now'. I also use this feature to build an index of Fish disks by putting the disks one at a time, into the external drive and re-executing the command:-

```
cat df1:Contents >>FishIndex
```

This only works with the more recent volumes where the disk contents are always described in a file named Contents. For older volumes you could try:-

```
cat df1:Readme.list* >>FishIndex
```

Be very wary though, append '>>' does NOT work with BCPL programs i.e. most of the original C: directory AmigaDOS commands. Why is BCPL so hard to get along with?

o Commands can be executed from a file using the internal command 'source' or by naming the file with a .sh suffix. If you type ManxInit as a command and the file 'ManxInit.sh' exists in your current or C: directory it will be automatically executed as a batch file.

o Function keys can be defined to any desired string using the in-built command 'set' for f1-f10 and shifted keys F1-F10. e.g.

```
set f1 "run sys:utilities/calculator"
```

The HELP key displays a list of all the available built-in commands in search order. All shell commands can be abbreviated and if cut too short to be unique the first command matching the name in this list is the one executed.

```
e.g. d = date
     di = dir
     dev = devinfo
```

o Csh also provides an internal 'path' variable separate from the one used by AmigaDOS. The internal path is searched first, before the command is passed on to AmigaDOS for execution.

o Command names or sets of commands can be given an 'alias' which is very useful for abbreviating a sequence of commands to be executed without creating a batch file. e.g.

```
alias ShowFiles "dir df0: ; dir df1:"
```

Aliases can also be used to give an existing command another name. e.g.

```
alias delete rm
alias type cat
alias makedir mkdir
```

These will set the internal commands to respond to the more familiar AmigaDos names also.

To enable you to set up your own command environment, Csh can be invoked from a CLI or the startup-sequence with a filename that will be executed first. Thus 'shell .login' will execute the series of commands from '.login' which may set your path, initialize the function keys and define your favourite aliases before passing control over to the command line prompt. Some examples of .login files are included on the disk and I have seen others included in various Fish disks.

There are many other commands and features of Csh that I have not discussed, I haven't used it frequently enough to become familiar with them all. I consider Csh an extremely useful extension to the AmigaDOS CLI environment and suggest that you try it yourself.

==

Metascope Review
by M. Griffiths

I was studying assembler at R.M.I.T. and desperately wanted to use my Amiga to do my assignments since we were studying the 68000 chip. All my early efforts were thwarted by an operating system that feels it has the right to move my programs wherever it likes in memory without telling me. Of all the cheek!

So dutifully I began to traipse around the stores asking for assistance in locating my missing programs but help was not at hand. The one comment that people made was:

GET METASCOPE!

What was this mysterious METASCOPE???

It's a pity I didn't find out in time for my assignments but as I have learned now, Metascope is an INDISPENSABLE programming aid if you have any ideas of programming the Amiga in assembler.

Metascope is run and then it loads a program and then the program is under its control. It can be stepped through, breakpoints can be set and run to, the registers are always displayed and can be altered and any number of windows can be opened to display sections of memory. In short, everything the assembly language programmer could want.

When Metascope begins it opens a single unassuming window entitled STATUS. This contains a dump of the registers, program counter, status bits and a disassembly of what the PC currently points at. From this point you can open any number of windows to display memory, breakpoints, hunks, and symbols. The memory windows can either show memory as code or as data. Also, they may be static and point at one address in memory or point to something like whatever SP points to. All windows may be labelled to allow you to keep track of them.

Breakpoints may be set and only triggered when they have been encountered a certain number of times. In the breakpoint window, the breakpoints are displayed along with how many times they have been encountered so far and how many times they need to be encountered before they are triggered.

Another very useful idea is to use Metascope to debug your C programs or to teach yourself assembler by writing C programs and then using Metascope to examine the disassembled code. To do this you simply compile your program with the -d option which tells the Lattice compiler to include debugging information. This makes all your symbols accessible toMetascope. You open up a symbol window, point to a symbol, and then open a memory window and Hey Presto! the memory window is open to that area of memory defined by the symbol. The program can now be run up to any of the symbols and then memory or registers examined.

This is a very brief idea of what the program can do but it really is a tremendously useful utility and also provides you with a means to understand more about your Amiga as you watch in detail as the code executes.

==

Short BASIC Routines From BaSIG

It's some God awful hour in the morning, my current project just went bush for the umpteenth time and I've run out of Mortein, so for something different to do I thought I would write something for the Workbench.

I would like to be able to present to you a small elegant routine that returns the answer to life, the universe and everything, however, all I can offer is a short delay routine that is system, not instruction dependent. This is useful if you wish to have a short delay in your programme that does not change when compiled. The delay involved can be varied by changing the variable Delay.

```
StoredTime = TIMER + Delay
Wate:
IF StoredTime > TIMER THEN Wate
```

If you have a small routine or short piece of example code that you think may be of interest to other AMigaBASIC users please contact me at a meeting or on (03) 375 4142. I hope, that with your help we (BaSIG) will be able to present a piece of code that will astound and astonish everyone each month. If your code is a little bit bigger than "short", we will still be interested as it can be included on a BaSIG public domain disk.

==

Some BASIC Ideas

by Mark Kelly

Here are some useful subprograms, tips and bug warnings for BASIC programmers.

SUB zzz

When debugging long, complex programs it is often handy to pause execution to examine screen output or take the opportunity to break execution. While a standard A$ = INPUT$(1) achieves this, it is annoying to have to type this in many times when tracing the source of a problem. ZZZ is a little subprogram that stops execution, prompts you to press a key, and - once you have hit a key - deletes the prompt before continuing. (The removal of the prompt is useful where the actual appearance of screen output is important.) To use the routine, simply add the subprogram to your program and put the command ZZZ wherever you want a pause in execution.

```
SUB zzz STATIC
     PRINT "Hit a key";
     a$ = INPUT$(1) : PRINT STRING$(9, 8);
END SUB
```

SUB ASK() and SUB sASK()

These subprograms enhance the standard INPUT command by offering the user an easy way to select default values. This gives the program flexibility (parameters can be easily altered) without annoying the user with interminable entry of values to be used for calculation.

A code fragment like...

```
start = 100
ASK "Start value?", start
```

results in...

```
Start value?  [RET=100]
```

The user can hit RETURN to use the current value of start (100), and the default value is printed after the prompt. Otherwise, a different value can be entered.

Different versions of the subprogram are necessary for different variable types. Below are ASK (for single precision or integer variables, depending on how the variable DEFAULT has been DEFined) and sASK for string input. Notice the semicolon after the INPUT statement - this suppresses the carriage return when the user hits RETURN. It allows ASK to print the default value after the prompt if the user just hits RETURN. The semicolon trick is well worth playing with. The AMIGABASIC manual merely shows it in the INPUT command's format description but does not explain its value.

```
SUB sASK(prompt$,default$) STATIC  'string ASK
     PRINT prompt$ " [RET=" default$ "]";
     INPUT ; " ", a$
     IF a$ <> "" THEN
     default$ = a$
     PRINT
     ELSE
     PRINT default$
     END IF
END SUB

SUB ASK(prompt$,default) STATIC    'numeric ASK
     PRINT prompt$ " [RET=" default "]";
     INPUT ; " ", a$
     IF a$ <> "" THEN
     default = VAL(a$)
     PRINT
     ELSE
     PRINT default
     END IF
END SUB
```

SUB SeeIF()

Another handy subprogram that finds its way into most of my programs - that's the beauty of subprograms: they can be copied into any code with no fears of corrupting variables already in use. A line such as...

```
SeeIf "Want to continue"
```

causes SeeIf to print the prompt message with a question mark and wait for the user to enter Y or N (uppercase or lowercase). Of course, it rejects bad input. SeeIf then sets the flags YES and NO according to the input and these flags can be used in easily-understood tests such as...

```
SeeIf "Want to print this"
IF no THEN RETURN
or
IF yes THEN GOSUB DoIt

SUB SeeIf(prompt$) STATIC
     SHARED yes,no
     PRINT prompt$ "?  ";
y:
     yn$ = UCASE$(INPUT$(1))
     yes = yn$ = "Y" : no = yn$ = "N"
     IF yes + no = 0 THEN BEEP : GOTO y
     PRINT yn$
END SUB
```

Something to try #1 -- A Novel Output Device

Ever thought about using menus for OUTPUT rather than INPUT? In one of my programs I had such a busy screen that I couldn't find screen space to show the results of the calculations. Where could I squeeze them? It was silly to open and close a window repeatedly just to peek at some figures. It was then that I realised the potential of the MENU commands. The following example should explain pretty clearly. Note that all values must be made into strings to be assigned as menu items.

```
report:
c = 1
MENU c, 0, 1, "RESULTS"
MENU c, 1, 1, "Section number" + STR$(p)
MENU c, 2, 1, "Across" + STR$(sx) + " to " + STR$(ex)
MENU c, 3, 1, "Down " + STR$(sy) + " to " + STR$(ey)
MENU c, 4, 1, "Maximum" + STR$(max)
MENU c, 5, 1, "Cuts " + STR$(cut& * vol) + " cu/m"
MENU c, 6, 1, "Fills" + STR$(fill& * vol) + " cu/m"
MENU c, 7, 1, "Ratio" + STR$(ratio!)
MENU c, 8, 1, "Excess" + STR$(xs&) + " cu/m"
IF fx THEN MENU c, 9, 1, "North/South gradient" +
 STR$(fx) + " " + ud$
IF fy THEN MENU c,10, 1, "East/West gradient" +
 STR$(fy) + " " + lr$
RETURN
```

After the values are calculated, just GOSUB REPORT and then, to see the results at any time, the user can just select the appropriate menu and there they are!

Something to try #2 -- An Alternative To Printing

Rather than sending copious output to the printer, which bogs down execution speed and eats expensive paper and ribbons, why not send output to a RAM: or VDO: file?

Sending output to memory rather than the printer is quick and you get a record that you can edit and save and even print out if you decide to. It's especially valuable when you're debugging a routine to format printed output. You can go through reams of paper before you perfect it. With RAM: output you can see exactly how the output would've appeared on paper.

To get started, use something like...

```
OPEN "O", 2, "ram:out"
 or
OPEN "O", devicenum, filename$
```

To send output, just use something like...

```
PRINT #2, a$; TAB(15); s(k);
```

(If you're debugging, you can easily redirect the output from RAM: to the printer by changing the device in the OPEN statement from "ram:filename" to "LPT1:")

Once the output is safely in the RAM: or VDO: file, you can either quit Amigabasic and get into the CLI or start up a CLI window without leaving AMIGABASIC (Use POPCLI from the Fred Fish disk collection. Without POPCLI you'll have to click the Amigabasic windows to the back and click the CLI icon on the Workbench disk). You can then TYPE the file to have a quick peek, ED the file to have a long look or change its format or enter:

```
RUN TYPE >PRT: filename
```

to print it as a background task while you type in ENDCLI (or click the CLI window to the back) and merrily return to AMIGABASIC. At last you can play with useful multi-tasking! If you decide the output is so valuable that you want to keep it, remember to COPY the ram: file to disk before shutting the computer down!

e.g. `COPY ram:out df1:BasicOutput`

Something to try #3 -- Multi-Device Output

What do you usually do when you want to both print something AND show it on the screen? Even worse, what if you want to try my idea above and ALSO send the output to a RAM: file? You could try:

```
LPRINT "The answers are" a/5 "and" tp/nd
PRINT "The answers are" a/5 "and" tp/nd
PRINT#2,"The answers are" a/5 "and" tp/nd
```

Pretty inefficient, isn't it? Try this instead...

```
OPEN"O",2,"LPT1:"
OPEN"O",3,"SCRN:"
OPEN"O",4,"RAM:OUT"
```

and when you send output, just use a loop...

```
FOR d = SD TO ED
     PRINT #d,"The answers are" a/5 "and" tp/nd
NEXT
```

In the example above you can select which devices are to be used. Set SD to 2 and ED to 4 to use all three. Set SD to 3 and ED to 3 to just send output to the screen. Set SD to 3 and ED to 4 for screen and RAM: output. You get the idea. You could send the output to every device under the sun if you were keen enough! Imagine feeding your printer, screen, modem and ram disk all at once!

Probably a Painfully Obvious Technique

Several times I've wanted to copy segments of code from one BASIC program to another. Originally I went through the tedious process of deleting all lines EXCEPT the desired segment, saving the segment in ASCII format, LOADing the program into which the segment was to be plugged, MERGEing the segment and then cutting it and pasting it into its proper position. It was then that I realised that I could simply COPY the desired segment (Right-Amiga-C), OPEN the destination Amigabasic program and PASTE the segment into the new program.

Of course you should save the original program if necessary before OPENing the new program. If you're shaking your head and muttering "That's flipping obvious! What an idiot!", I apologise. Until I made my big discovery I assumed that cutting and pasting could only be done within the same text. In fact, cut or copied text is even retained if Amigabasic is QUITted and started up again. This applies equally well to TEXTCRAFT! To copy bits from one document to another, open the source file, COPY the text to be moved, OPEN the destination document, find the right place and PASTE it in.

AmigaBASIC Blues -- Some Problems And Some Fixes

(a) When you create a program that uses a 15-colour screen (e.g. SCREEN 2,640,200,4,2) and try to use menus, you may find that the highlighting colour for selected menu items is something like brilliant yellow so the text of the highlighted menu item is nearly invisible. The culprit is COLOR 14 which is used as the background colour when highlighting. To return it to a nice, dark shade use PALETTE 14,0,0,0

(b) "Runaway" list window scrolling. You've held the cursor key down too long and the line you wanted to see in the list window has appeared and keeps scrolling past your eyes and disappears again. Frustrating! To immediately stop list window scrolling, just click anywhere in the output window.

(c) Two problems I haven't solutions for. Firstly, has anyone had trouble with the SAY command when using a hi-res screen? I tried getting my Amy to talk to me from a 15-colour screen and she sounded like a raspy, garbled alcoholic. Any ideas? The other problem I've cursed and sworn over a lot is cutting text with right-Amiga-X. Five times out of ten the text disappears, surely enough, but it disappears entirely! A plaintive "x" appears where the text was, and no amount of pasting in the world will retrieve the text. Using the menu to cut works fine, but using the keyboard shortcut is so much of a gamble that I always take the precaution of saving the program to RAM: before risking a cutting manoeuvre. Has anyone else suffered this?

Hope the hints are helpful. Happy hacking!

==

Executing CLI commands from AmigaBASIC

by Carolyn Scheppner
CATS - Commodore Amiga Technical Support

[Editor's note: this item found its way to us via the magic of the world-wide computer network, USENET]

Here's something useful for Basic programmers. It shows one way to use the AmigaDOS Execute() function to execute CLI commands from within an AmigaBASIC program.

This example redirects the output of the CLI Dir command to a ram:temp file, which can then be OPENed in Basic for PRINTing the output to any AmigaBASIC window.

It requires a dos.bmap created with the 1.2 Extras version of ConvertFD.

```
REM - DirToFile.bas Do a Directory to a Ram File
REM C.  Scheppner CBM 01/88

Main:

REM - Functions from dos.library
DECLARE FUNCTION Execute& LIBRARY
DECLARE FUNCTION xOpen& LIBRARY
```

```
PRINT:PRINT "Looking for bmaps ...  ";
LIBRARY "dos.library"
PRINT "found them."

nil$ = "NIL:" + CHR$(0)
nhandle& = xOpen&(SADD(nil$),1006)

InLoop:
PRINT
INPUT "Directory name (or <RET> to quit): ",dirname$
IF dirname$ = "" THEN GOTO MCleanup
PRINT

com$ = "dir >ram:temp "+ dirname$ + CHR$(0)

PRINT ">>> Executing DIR > ram:temp"
PRINT

success& = Execute&(SADD(com$),0,nhandle&)

PRINT ">>> Reading and printing ram:temp file"
PRINT

OPEN "ram:temp" FOR INPUT AS #6
WHILE NOT EOF(6)
 INPUT#6,a$
 PRINT a$
WEND
CLOSE #6

GOTO InLoop:

MCleanup:
CALL xClose&(nhandle&)
LIBRARY CLOSE
END
```

==

RJ Mical's ProSuite

by M. Griffith

Here is an opportunity for the beginning C programmer on the Amiga to introduce some very professional routines into their programs, shorten program development time, and learn about C programming in the best possible way. Prosuite is a collection of routines for the Amiga written by RJ Mical and placed in the public domain. They are very well documented, and several example programs using them are included.

Prosuite contains a collection of five routines and the promise of more to come as RJ builds up his own public domain library for others to use. The routines are SetWait, XText, ColorWindow, DoRequest, and FileIO.

SetWait is the simplest of the functions in that it allows the programmer to change the pointer that is displayed while the Amiga is busy loading programs etc. (ie. the wait pointer) from it's usual snoring cloud.

XText is a collection of functions to allow very fast text rendering in your programs. It achieves this speed at the expense of flexibility. A very limited number of fonts are available along with a very limited number of styles.

ColorWindow is a function which when included gives the user a mechanism for changing the screen colors of your programs.

DoRequest is a collection of functions to simplify the production of requesters in your programs. You initialize a ReqSupport structure and call the DoRequest function and up pops the requester.

Finally, FileIO is a series of routines to create a file selector for loading and saving files from within your programs. Again you initialize a support structure and call the function which creates a file requester and monitors the users choice for you.

I'm a beginner when it comes to programming the Amiga in C, but I found these routines to be very easy to include in my programs. They are very well documented and the source code is well commented. They certainly made my programs look much more professional than they are. I would recommend to any beginner in C to obtain them from the Fred Fish disks and have a look at them. It will be well worth the trouble.

==

Power Windows 2.0

by M. Griffith

Power Windows 2.0 is billed as a product for "power users". I am definitely NOT a power user and yet have found this an enormously educational piece of software. By freeing the beginning programmer from the trauma of having to generate the MANY structures concerned with programming the Amiga, it frees you to concentrate on the programming side of things. At the same time, by creating these structures for you, the program allows you to study these structures and thus come to better understand how they are used.

Power Windows 2.0 comes on one disk with a small, 11 page manual. This may seem small for a manual but it contains all the necessary information and the program is better learned through using it, rather than reading the manual. One thing the manual does not do is give details about the Intuition interface so if you don't know much about it then you had better have a good book. The disk is unprotected.

The basic idea with Power Windows 2.0 is that you "draw" on the screen the windows you wish, size them, place them on the screen where you want them and then Power Windows 2.0 creates all the necessary structures to reproduce them in a program.

It is capable of producing code for Lattice, Manx, assembler and TDI Modula2.

The screen may be of any resolution and any (appropriate) number of bit planes. It allows you to attach menus to windows, create and attach gadgets, and place text anywhere on the windows. The rendering for the gadgets maybe loaded from any IFF brush as used by programs like Deluxe Paint. The structures for all of these are also created.

The program is fully menu driven, although there are keyboard shortcuts for many of the menu items. To begin, you first select Define Screen Type to determine whether you are using the WorkBench screen or a custom screen. When this is done, all windows created will use this screen. The screen selected

APAL for the AMIGA

BY COMMODORE

FULLY FEATURED 2 - DIMENSIONAL CAD

BASFORD SYSTEMS
25 Neale Street
Bendigo,
Victoria, 3550
Ph.(054) 43 3349

DRAWING FEATURES

Help
APAL is a command driven program and context sensitive help is provided via the Amiga Help key.

Lines
Line drawing commands follow a well structured syntax and provide over one thousand different ways to specify the geometry of a line. Special syntax is provided for Arrow head, Leader and Survey lines.

Curves
Like lines, many options are provided to fit circles and arcs to the required situation.

Text
Extremely flexible Text commands allow such things as any combination of horizontal and vertical justification of single as well as multi - line blocks of text. Multiple and user defined fonts are allowed in the one drawing.

Zoom
Comprehensive windowing options allow rapid redrawing and removal of drawing elements, as well as zoom in and out.

Texture and Thickness
Drawings may be done in up to nine different line thicknesses and up to six different textures. Texture patterns are user definable as well.

Layering
Drawings can be entirely on one layer or may be spread over up to thirty - two layers.

Cross hatching
Powerful easy to use cross hatching commands allow shading of simple and complex areas (including cut - outs) with user control over spacing, angle and starting point.

French curves
Smooth curves may be generated through a series of points.

Mouse digitising
The mouse may be used to digitise existing sketches and diagrams to any scale in either point or continuous mode.

Symbols
Any drawing can be used as a symbol and any drawing can be used in another drawing via scaling, rotation and translation options (others too). Transformations can be automatic by specifying two points, between which the incoming drawing must fit.

Startup file
Many drawing parameters take their initial value from the users startup control file. This allows the user to customise APAL to their needs.

Relocation
Parts of a drawing can be shifted around or copied while performing transformations such as scaling and rotation at the same time.

Grid
For drawings with much regularity, a grid can be used to "round off" drawing commands to the nearest grid point.

Deletion
Extensive commands are available for deleting individual drawing elements, such things as lines, curves,text, chains of lines, sub-pictures, etc. As well arbitrarily shaped areas of the drawing can be removed or isolated.

Amiga features
Just two of the many features of the Amiga supported by APAL are, user definition of the function keys and Amiga Menus.

Plotting
A weakness in many products but not APAL. Extensive options for plotting are provided. For instance lines can be thickened correctly, even on single pen plotters. Plotting by layers or thickness are just two other options.

Documentation and Support
APAL is provided on an Amiga standard 3 1/2 inch disc. It comes with a two hundred plus page manual and quick reference card. In the manual is a registration form so that users can be notified of updates (and there will be some) and tips. As the product is designed in Australia, local support is assured and feedback from users is encouraged directly to Basford Systems.

Applications
APAL is suitable for many draghting jobs from simple sketches to major technical works. Its major feature is its ease of use and hence productivity.

Compatibility
APAL is for the most part keystroke compatible with the leading minicomputer CAD system (VAX, Apollo), Palette. File interchange with Vax based Palette systems is even possible.

Requirements
APAL requires a Commodore Amiga model 500, 1000, or 2000. 512k is required although more can be used. AmigaDos 1.1 or 1.2 can be used with just the internal 3 1/2 inch drive.

Sample drawings
Several sample drawings are shown overleaf to illustrate the capabilities of APAL. Note that they are not to scale shown,but were originally and were simply scaled by the software for 1:1 output and reproduction here.

$ 199.00

MADE IN AUSTRALIA

NOW AVAILABLE AT THESE LOCATIONS;

ULTRAPHASE • C • O • M • P • U • T • E • R • S • ULTRAPHASE • C • O • M • P • U • T • E • R • S •

GREYTHORN:	296 Doncaster Rd.	(03) 857 4233
MELBOURNE:	Hub Arcade, 318 Lt Collins St.	(03) 277 7660
KNOX :	5 Little Burwood Hwy.	(03) 887 1900
OAKLEIGH :	52 Atherton Rd.	(03) 568 7944

will also appear as the backdrop for all subsequent work. If you want to design your own palette then Screen Palette is selected next.

When the screen is ready you select Open New Window and a small window will appear on your screen. By resizing this and moving it you can have exactly the right look for your program. Selecting Edit Window Characteristics allows you to alter all the IDCMP flags, the pens used, window title, window flags etc. When you exit from this the window will appear EXACTLY as it will appear in your program. This allows you to experiment with the window flags and see what difference they make to the window's appearance.

After this, as already stated, you can add menus or gadgets and thus turn your window into any sort of custom window or gadget you like. Finally when the code is generated you have the option to generate all the code or only parts like the palette, menus, windows etc. One useful selection is the generation of the event handler which will then handle all menu selections for your program. This is very useful for the beginner.

There are many examples included on the disk including a skeleton program for Lattice which will allow you to get a program up and running quickly and also provide another learning tool for you to examine.

In conclusion, Power Windows 2.0 is an extremely useful programming utility for the beginner in programming the Amiga Intuition interface and I suspect a very powerful productivity tool for the advanced programmer.

===

Developer's Corner

by Chris Tremelling

Interest in the idea at the last Developers Sig meeting has resulted in the following list of top ten Public Domain programs (as used by members of the Sig).

The programs are varied but they have one thing in common - they improve the user's environment. Although this list was compiled by developers, the programs below may be useful to others.

1. Conman

   Conman is a replacement console handler that provides line editing and command line histories completely transparent to any application program that uses CON: windows. (Especially useful for two finger typists - it allows you to correct any mistyping without retyping the entire line.)

   Version 1.0 is available on Fish disk 100 with earlier versions also on 90, 81 and 69. Amigan disk 11 and 9 also contain copies.

2. PopCli

   Provides a simple way of starting another CLI at any time without having to load workbench or exit whatever program you may be using. Also has a built in screen saver mode that automatically blanks the Amiga console screen when there has been no input.

   Version 3 is available on Fish disk 84, Amicus disk 12 and 15 or Amigan disk 4 and 10.

3. Blitz Fonts/Blitz

   Blitzfonts makes text output up to 6 times faster and comes with a blindingly fast text viewing utility, Blitz.

   These programs are available on Fish disk 60 or Amigan disk 9. WarpText is a library containing a text output routine that allows you to create programs which output to the screen in the blink of an eye, and it can be found on Fish disk 96.

4. ASDG Recoverable Ram Disk (VDO)

   The ASDG-RRD is a ram disk which unlike the standard AmigaDOS ram disk can survive warm boots and most Guru's. Very useful with extra memory.

   The recoverable ram disk is available on Fish disk #58, but it also seems to come with almost every Amiga memory expansion board.

5. SunMouse

   Sunmouse makes your mouse behave like a Sun Microsystem Sunwindows mouse. No longer will you have to 'click' in a window to make it active, just move the mouse pointer into the window and start typing.

   Version 1.0 of Sunmouse is available on Fred Fish disk 65 and Version 1.0 HeliosMouse (another version of the Sunmouse) is available on Fish disk 111 and 94.

6. Make

   A utility for compiling and linking program modules. It checks the creation dates of the different modules, recompiling only those that are out of date.

   Available on Fred Fish disk 45 and Amigan disk 4.

7. DirUtil

   DirUtil is a utility for wandering around a directory tree with the ability to perform various operations on files.

   A number of different versions are available, DirUtil can be found on Fish disk 49, Amicus disk 11 and DuM2 is on Fish disk 75.

8. Undelete

   Undeletes a deleted file, available on Amicus disk 12.

9a. DME

Dme is a simple WYSIWYG editor designed for programmers. Its features include arbitrary key mapping, fast scrolling, title-line statistics, multiple windows and more.

Version 1.27 is available on Fred Fish disk 93, older versions can be found on 87, 74 and 59.

9b. EMACS

EMACS, another editor available on an wide variety of computers (ie: Vax, IBM ...) can also be used on Amigas.

It can be found on Fred Fish disks 119, 93, 68, 61 and Amigan disk 1 in a number of forms.

10. File Zap

A utility that can be used to examine and modify any sort of file.

Available from Fred Fish disk 14 and Amicus disk 16. NewZap, a similar program, is available on Amigan disk 7.

Let me know how you like our Top 10 - comments welcome.

Developers SIG Members Please Note

A tutorial on using "IMAGE Gadgets" will be presented at the next meeting as well as a demonstration using "Power Windows". NEWCOMERS WELCOME!

FACC Review
by Chris Tremelling

Facc (Floppy Accelerator) is a disk caching program which can dramatically increase floppy disk throughput. Although similar to "Addbuffers", Facc offers a number of advantages:

i The number of buffers can be increased or decreased with the touch of an icon at any time.

ii Facc's buffers can use fast memory, whereas the AmigaDOS utility "Addbuffers" can only use chip memory.

iii Buffers are dynamically directed to the busiest disk.

iv If memory becomes very low, Facc can be closed. Once Addbuffers has been run it can't be deleted until a reboot.

A disk cache is an area of memory used for buffering disk blocks. If a program needs to read information from disk and that information is already in the cache, then the in-memory copy is returned, which means the disk doesn't have to be read. If this information is not in the cache, then it is read from disk and a copy kept in the cache. Successive reads of that information are nearly instantaneous. Copying from the buffer is a lot faster than copying from disk.

When a block is written to disk, a copy is also kept in the cache for later reuse. Each disk block in the cache occupies one buffer. These buffers are reused according to the LRU (Least Recently Used) algorithm.

The effectiveness of disk caching can only be realized when the data being accessed is already in the cache. If the cache is too small, blocks are aged out before they can be re-read. The user can adjust the size of the cache until effective caching is achieved. Facc provides a small screen with statistics which allows you to measure this effectiveness.

Using the Word Perfect spelling checker as a guide, the time required to check the spelling in a 600 word document was measured with different cache sizes.

Note: All times included loading the cache, and each buffer is 544 bytes.

Cache size:	Time:
0	2 minutes (No Facc)
150	2 minutes (Caching not effective)
300	1 minute 18 seconds
450	1 minute

From this table we can see that the larger the cache, the greater the time saved. With larger documents and cache, the savings would be even greater (eg: 6100 word document in 7 minutes).

While a ram disk may be even faster, Facc avoids the need to load up a ram disk and can reduce the wear and tear on floppy drives.

The advantages and ease of use of Facc make it worthwhile for any application that heavily uses the floppy disk drives.

[Editor's note: I've read what's written on the Facc box, and it seems to imply that it only works for floppy disks. Can anyone tell me if it will cache hard disk sectors?]

Pssst!! Want Another 1/2 Meg Ram for Peanuts?
(subtitled: How to Test your Nerve)
By Robert Glucz

Spurred on by the incessant coaxing from our esteemed editor for somebody to contribute something (anything) printable, I have put together a review of of an internal hack which expands an A1000 to 1 Megabyte. The hardware project to which I refer was published in Amazing Computing Vol.2 No.1, and runs to some nine pages in the magazine. [Editor's note: this issue is available from our club library.]

As an introduction, I would say I am a typical frustrated Amiga user - all those add-ons and no money to buy them! I am currently striving to understand the intricacies of C and the constant waits while the compiler, assembler, linker, editor, etc. load from disk is extremely frustrating; so, when I first saw this article I thought,

"Hallelujah!!".

Before my natural cynicism stepped in (where's the catch?) I raced out and bought all the listed components. This required a fair amount of letting-my-fingers-do-the-walking to find the DRam chips (non-standard types, thanks Commodore!) but eventually they were located at a reasonable price.

Now, with all the parts laid out on my desk, I tackled step 1 - read the article; then step 2 - read the article again; then emboldened by my success I tackled step 3 - dive right in! All caution to the wind, off came the lid (I love all those signatures molded into the top cover - tempted to scratch mine in there after this mod). Pretty bare circuit board, I thought, as I removed the Kickstart Daughter Board. (If this idea scares you, stop reading now - it gets worse!). A precis of the procedure to be followed next would read something like this; solder sockets to the existing ram chips; solder two DRams together "piggy-back" style; insert them into the sockets; then run wires from their chip select pins to existing decoder chips on the mother board. Soldering directly onto the mother board and snipping pins on existing chips is all par for the course, so faint hearts need not apply. This entire surgery was performed in approximately four hours from go to whoa, and is reasonably straightforward.

At this point I had recabled everything to my Amiga, with the lid still off, (I wasn't that confident!) when it suddenly hit me, WhatIfItDoesn'tWork? "What have I done?" I thought, as I nervously flicked the power switch from arms length. No smoke, Phew! Kickstart OK, Workbench OK, Addmem .... OK!!! Rslclock says I've got over 700k at the workbench - you beauty, pat, pat.

Hang on, let's see if it actually does anything; copy df0: to ram:, chug, chug, KERASH. Reboot. Let's try that again; chug, chug, you guessed it - Wiley Coyote, MUD. (in joke) Step 4 - check your work; aha, a chip select wired to the wrong pin. Suffice to say, the project is quite simple with no obviously difficult pitfalls along the way (he says, now that it works).

Seriously though, the article is extremely well documented with photos of the board layouts, circuit diagrams of the modifications and excellent step by step instructions. I would list the requirements of this project to be; fairly good competence in soldering techniques, general home workshop equipment (e.g. sidecutters, soldering iron, etc.); and a liberal dash of 'Who cares if I stuff it up'.

Now the important part, my wallet was lightened to the tune of $140 for the total parts list required, which I consider good value for the convenience of the extra 512k ram. Commercial units retail for at least $450.

The design does not provide auto-configure memory (what do you want for $140?). It does work fine with all programs that can use expanded memory, that is, all the programs I've tried so far, with the notable exception of the ASDG recoverable ram disk (probably due to the addresses it occupies - this should be fixable, I'm working on it now, so see a later article - what have I let myself in for?)

I have been running this mod for about two weeks now, with no problems whatsoever, and I would heartily recommend it to anyone with the appropriate aptitude to build it and a lack of cash for a commercial unit.

If anyone has any questions/tests to try/suggestions I can be contacted (at odd hours) on 763-9552.

PHANTASIE III (The Wrath of Nikademus)
By Nigel Harwood

Phantasie III by SSI Inc is quite a good game for around $73. That's assuming you like D & D type games. As you may have realised by my other short reviews, I do.

As I would have expected from a mature company like SSI, the game is very well thought out and contains some really great features. Your task is to defeat the enemy by the name of Nikademus. To do this, you have to build up strong enough characters to do this by fighting your way through the miriad of monsters (80 types!).

The playing field is a whole continent much the same as in the "Faery Tale" game, except that here you don't have to spend three hours walking between towns. The towns, from where you base your expeditions, are scattered strategically over the continent and its smaller islands.

During game play, you do the normal sorts of things like buying equipment, restoring your party by sleeping in the inn, etc. Unlike "Bard's Tale", you can save your game to disk at any town you reach, and the banks in each town must be part of a network, because money you put in the bank in one town is always accessible in all of the others.

There is the usual lot of weapons, shields, armour, spells, potions, scrolls, member races, member classes and there is even a social class associated with each member.

When you leave some of the towns, a dungeon of some sort can usually be found nearby, if you can reach it before getting clobbered by monsters, that is. The dungeons operate a little like the ones in "hack" or "larn" ie with paths and doors appearing as you move along. When you enter a dungeon, you can elect to load a map saved the last time you were here, hence you never have to get the pencil and paper out and map out the dungeons (I imagine some may like to do this, I don't).

There are four planes of existence in the game, the "material plane" which is the normal one, the "plane of light", the "plane of darkness" and the "netherworld" (where of course Nikademus can be found). As yet I haven't figured out how this is all going to pan out, but don't tell me, I'll enjoy finding out ;-)

I suppose I have raved about this game for long enough, it just seems so full of interesting things, and I haven't even mentioned that the fighting mode is really good if a little long winded.

One big black mark against SSI is the manual. The game was written for a multitude of computers and they must have taken ALL the instructions for all of them and thrown them into the manual. They have a seperate section on Apple][specifics but they also put a lot of instructions and information for other computers all through the manual which don't relate to the Amiga version (most confusing).

Finally, the disk is not copy protected and I was shocked by this until I realised that there must be manual type protection. But no, I started the game up and played for about twenty minutes and nothing, then, when I had found some great bit of treasure, it asked me a question about a particular page of the manual. Not a particularly nasty form of protection, I guess.

===

PAL Video Modification

by Brian Parker

[Editor's note: This article has been reprinted from the the March 1988 issue of Cursor, Amiga Edition, Newsletter of The Commodore Computer Users Group (QLD) Inc.]

This article describes a hardware modification to version 1.3 Amigas (which have a PAL AGNUS chip, but NTSC composite video circuitry), which will give full PAL composite video. This is most useful for recording the Amiga output on video.

Parts List

33pF, 56pF, 10pF and two 220pF ceramic capacitors
0.001uF greencap
9.8 - 60pF trimmer capacitor (or something around 30pF)
47k 1/4 watt resistor
4.433619 Mhz PAL crystal
(Optionally, a 28.37516 Mhz oscillator from a Commodore repair centre)

Procedure

Disassemble the machine. Locate the MC1377 chip at the top left corner (looking from the front). Cut the tracks leading to pins 1 and 17 on the underside of the board, and disconnect pin 20 from earth by cutting the tracks on the component side of the board. (N.B. It is connected to earth on both the inside and outside of the pin. Cut the inner track by levering a small screwdriver across it several times.)

Locate C45 and replace it with the 33pF ceramic capacitor, then locate C54 and replace it with the 56pF capacitor. Now connect the other components as follows:

1. Solder a 220pF ceramic capacitor across pins 17 & 18.
2. Solder the other 220pF ceramic capacitor across pins 15 & 18.
3. Solder the 47k resistor across pins 1 and 16.
4. Solder the 0.001uF greencap from pin 1 to the adjacent earth plane.
5. Solder the 10pF ceramic across the trimmer capacitor then solder one endof the trimmer to pin 15.
6. Solder the crystal from pin 17 to the other end of the trimmer.
7. Solder a wire from the can of the crystal to a nearby earth.

When doing the above, try to keep the component lead lengths as short as possible. If you are not into free-form modern sculpture, you could possibly de-solder the MC1377 and mount the whole lot on a bit of veroboard, with connections to the original IC position.

Once this is completed, partially reassemble the machine - leaving the top shield off. Turn the Amiga on, connected via a suitable cable to the composite input of your monitor. Let the machine warm up, then adjust the trimmer until interference lines (seen most prominently at the border of orange and blue) are stationary and minimized. Then complete reassembly.

One last modification is needed to bring the machine up to full PAL standard. This is to replace the main oscillator with a 28.37516 Mhz version. As this is only 1% different to the NTSC clock, it would probably only need to be changed if something like genlock was used.

===

Editor's Column

Quite a number of people have asked me about my Spirit internal 1.5 Mbyte memory board. The questions are usually where did you get it and how much did it cost. Since I had been saving memory chips for quite a while, I wanted to buy a board without memory. I couldn't find anyone locally who had a board I liked, so after seeing adverts in the overseas magazines for the Spirit board, I rang the USA to get prices. The 1.5 Mbyte board with no memory was a few cents under US$300, with $15 or so for airmail postage. After thinking about it for a few days, I rang back and ordered one. Two days after that, I picked up a copy of "The Australian Commodore and Amiga Review" in a newsagents and saw an advert for what I'd just ordered for $450, within a few dollars of what the US$300 would be when converted. Bum, I thought. If customs stops it, it'll be dearer than buying it locally.

Well, customs let it through, and my MasterCard has now been debited $440.34, pure luck that it didn't cost me any more. Anyway, everyone keeps asking me the name or details for the company that advertised the board locally. The unfortunate answer to that question is that not only can't I find the advert, but the latest [Feb '88] issue came out today and they haven't advertised in it! So, unless a reader knows who these people are (I believe it was a Sydney company), you'll have to chance getting one from the USA. That's pretty bad, because this local company is giving buyers a very good deal, unlike the huge rip-offs that some other companies are into.

Well, that's all I've got room for this month. The March meeting is still on a Saturday, the 12th to be precise. For April through to the end of the year (except for November), we've managed to change to the third SUNDAY, just like everyone wants it! See you.

Newsletter Back Issue Order Form

Issue Numbers:							
Number of issues at $2 each						$	
Less 10% for 5 or more						- $	
Club Use Only						Total $	

Mail to: Amiga Users Group, PO Box 48, Boronia, 3155

Member's Name:

Address:

SOFTWARE ORDER FORM

Disk numbers :							
Disks supplied by Amiga User Group @ $8						$	
Disks supplied by member @ $2						$	
Club Use Only Receipt #: Mailed on: / /						Total $	

Mail to: Amiga Users Group, PO Box 48, Boronia, 3155, Victoria.

Member's Name:

Address:

Application for membership of The Amiga Users Group Inc

Membership is $20 per year. Send your cheque to: **Amiga Users Group Inc, PO Box 48, Boronia, 3155**

Surname: ____________________

First name: ____________________ (no initials)

Address: ____________________

____________________ Postcode: ________

Phone Number: ____________ STD Code: ______

What services would you like AUG to provide: ________

____

____

Signed: ____________ Date: ________

Details on this side are optional

Year of birth: ________ Which model Amiga: ______

Occupation: ____________________

Interests: ____________________

____

____

Dealer's Name: ____________________

Dealer's Address: ____________________

____________________ Postcode: ______

In the event of my admission as a member, I agree to abide by the rules of the Association for the time being in force.

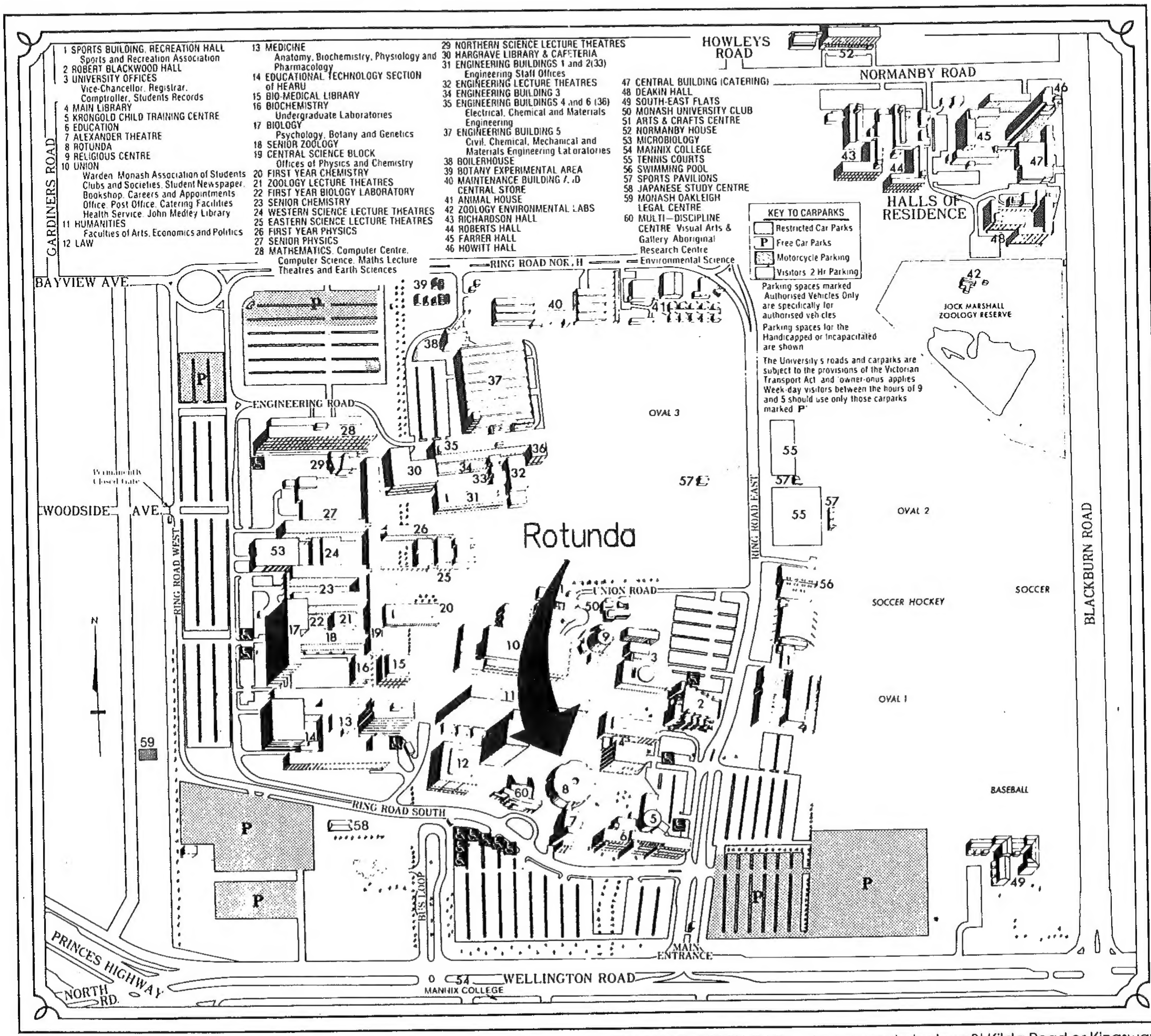
Club Use Only	Date	Paid	Rcpt #	Memb #	Card Sent

March 1988 Amiga Workbench

Don't Forget

We now meet in the Rotunda, Monash University, Wellington Road, Clayton

Monash University is in Wellington Road, Clayton. See Melways Map 70, reference F10. Melways map 84A shows the University Campus in details. I've drawn a huge arrow on the map below to show where the Rotunda is. The best place to park your car is the car park area between Wellington Road and the Rotunda. The entrance to the Rotunda is virtually at the point of the arrow.



1 SPORTS BUILDING, RECREATION HALL Sports and Recreation Association
2 ROBERT BLACKWOOD HALL
3 UNIVERSITY OFFICES Vice-Chancellor, Registrar, Comptroller, Students Records
4 MAIN LIBRARY
5 KRONGOLD CHILD TRAINING CENTRE
6 EDUCATION
7 ALEXANDER THEATRE
8 ROTUNDA
9 RELIGIOUS CENTRE
10 UNION Warden Monash Association of Students Clubs and Societies, Student Newspaper, Bookshop, Careers and Appointments Office, Post Office, Catering Facilities Health Service, John Medley Library
11 HUMANITIES Faculties of Arts, Economics and Politics
12 LAW
13 MEDICINE Anatomy, Biochemistry, Physiology and Pharmacology
14 EDUCATIONAL TECHNOLOGY SECTION of HEARU
15 BIO-MEDICAL LIBRARY
16 BIOCHEMISTRY Undergraduate Laboratories
17 BIOLOGY Psychology, Botany and Genetics
18 SENIOR ZOOLOGY
19 CENTRAL SCIENCE BLOCK Offices of Physics and Chemistry
20 FIRST YEAR CHEMISTRY
21 ZOOLOGY LECTURE THEATRES
22 FIRST YEAR BIOLOGY LABORATORY
23 SENIOR CHEMISTRY
24 WESTERN SCIENCE LECTURE THEATRES
25 EASTERN SCIENCE LECTURE THEATRES
26 FIRST YEAR PHYSICS
27 SENIOR PHYSICS
28 MATHEMATICS, Computer Centre, Computer Science, Maths Lecture Theatres and Earth Sciences
29 NORTHERN SCIENCE LECTURE THEATRES
30 HARGRAVE LIBRARY & CAFETERIA
31 ENGINEERING BUILDINGS 1 and 2(33) Engineering Staff Offices
32 ENGINEERING LECTURE THEATRES
34 ENGINEERING BUILDING 3
35 ENGINEERING BUILDINGS 4 and 6 (36) Electrical, Chemical and Materials Engineering
37 ENGINEERING BUILDING 5 Civil, Chemical, Mechanical and Materials Engineering Laboratories
38 BOILERHOUSE
39 BOTANY EXPERIMENTAL AREA
40 MAINTENANCE BUILDING AND CENTRAL STORE
41 ANIMAL HOUSE
42 ZOOLOGY ENVIRONMENTAL LABS
43 RICHARDSON HALL
44 ROBERTS HALL
45 FARRER HALL
46 HOWITT HALL
47 CENTRAL BUILDING (CATERING)
48 DEAKIN HALL
49 SOUTH-EAST FLATS
50 MONASH UNIVERSITY CLUB
51 ARTS & CRAFTS CENTRE
52 NORMANBY HOUSE
53 MICROBIOLOGY
54 MANNIX COLLEGE
55 TENNIS COURTS
56 SWIMMING POOL
57 SPORTS PAVILIONS
58 JAPANESE STUDY CENTRE
59 MONASH OAKLEIGH LEGAL CENTRE
60 MULTI—DISCIPLINE CENTRE Visual Arts & Gallery Aboriginal Research Centre Environmental Science
KEY TO CARPARKS
Restricted Car Parks
P Free Car Parks
Motorcycle Parking
Visitors 2 Hr Parking
Parking spaces marked Authorised Vehicles Only are specifically for authorised vehicles
Parking spaces for the Handicapped or Incapacitated are shown
The University's roads and carparks are subject to the provisions of the Victorian Transport Act and 'owner-onus' applies. Week-day visitors between the hours of 9 and 5 should use only those carparks marked P
Rotunda
HOWLEYS ROAD
NORMANBY ROAD
HALLS OF RESIDENCE
GARDINERS ROAD
BAYVIEW AVE
WOODSIDE AVE
ENGINEERING ROAD
RING ROAD NORTH
RING ROAD WEST
RING ROAD EAST
RING ROAD SOUTH
UNION ROAD
BUS LOOP
MAIN ENTRANCE
WELLINGTON ROAD
MANNIX COLLEGE
PRINCES HIGHWAY
NORTH RD.
BLACKBURN ROAD
JOCK MARSHALL ZOOLOGY RESERVE
OVAL 1
OVAL 2
OVAL 3
SOCCER HOCKEY
SOCCER
BASEBALL
N

BY PUBLIC TRANSPORT . . . The simplest method is to take a train from Flinders Street or Loop stations on the Dandenong/Pakenham line to either Huntingdale or Clayton. Buses run from these stations to the campus or there is a taxi rank at Clayton. With suitable connections the trip takes about 45 minutes— but it can take longer! An inner neighborhood ticket will take you all the way via Huntingdale station and the bus, but you will need to purchase a comprehensive ticket for the trip via Clayton, which encompasses two neighborhoods. The campus is also served by buses from Box Hill, Blackburn, Belgrave, Chadstone, Jells Park-Glen Waverley, Dandenong-Mulgrave, Oakleigh and Elwood.

FROM THE CITY BY CAR . . . An easy route is along St Kilda Road or Kingsway/ Queens Road and then on to Dandenong Road. The campus's tall Menzies Building comes into view a kilometre or so before the left turn into Wellington Road on which the main entrance is located. Allow 40-50 minutes for the trip. Drivers should note that restrictions apply in some car parks weekdays 9 a.m. to 5 p.m. and fines **do** apply. There is ample unrestricted parking and, closer to buildings, designated two hour visitor car parks — check the map or ask at the Gatehouse.